**ASSIGNMENT**

1. Create a class FLOAT that contains one float data member .Overload all the four

arithmetic operators so that they operate on the objects of FLOAT.

Code:

#include<iostream>

using namespace std;

class FLOAT {

private:

float distance;

public:

FLOAT(float d = 0.0) : distance(d) {}

// Overload + operator

FLOAT operator+(const FLOAT& f) const {

return FLOAT(distance + f.distance);

}

// Overload - operator

FLOAT operator-(const FLOAT& f) const {

return FLOAT(distance - f.distance);

}

// Overload \* operator

FLOAT operator\*(const FLOAT& f) const {

return FLOAT(distance \* f.distance);

}

// Overload / operator

FLOAT operator/(const FLOAT& f) const {

if (f.distance == 0.0) {

throw "Division by zero condition!";

}

return FLOAT(distance / f.distance);

}

// Getter function

float getDistance() const {

return distance;

}

};

int main() {

FLOAT f1(30.0), f2(20.0);

FLOAT f3 = f1 + f2;

FLOAT f4 = f1 - f2;

FLOAT f5 = f1 \* f2;

FLOAT f6 = f1 / f2;

cout << "f1 + f2 = " << f3.getDistance() << endl;

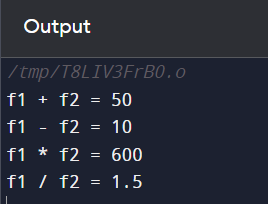
cout << "f1 - f2 = " << f4.getDistance() << endl;

cout << "f1 \* f2 = " << f5.getDistance() << endl;

cout << "f1 / f2 = " << f6.getDistance() << endl;

return 0;

}



2. Define a class string. Overlaod ==operator to compare 2 strings.

Code:

#include<iostream>

#include<cstring>

class String {

private:

char\* str;

public:

// Constructor

String(const char\* s) {

str = new char[strlen(s) + 1];

strcpy(str, s);

}

// Destructor

~String() {

delete[] str;

}

// Overload == operator

bool operator==(const String& s) const {

return strcmp(str, s.str) == 0;

}

// Getter function

const char\* getString() const {

return str;

}

};

int main() {

String s1("Hello");

String s2("Hello");

String s3("World");

if (s1 == s2) {

std::cout << s1.getString() << " is equal to " << s2.getString() << std::endl;

} else {

std::cout << s1.getString() << " is not equal to " << s2.getString() << std::endl;

}

if (s1 == s3) {

std::cout << s1.getString() << " is equal to " << s3.getString() << std::endl;

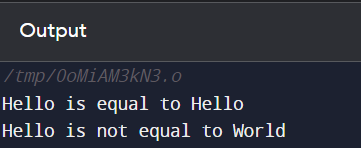
} else {

std::cout << s1.getString() << " is not equal to " << s3.getString() << std::endl;

}

return 0;

}



3. Create a Complex class that has real(int) and img(int) as member data, and has getData

and showData functions. Then also overload the following operators for Complex class. =,

==, +, ++, --,

Code:

#include<iostream>

using namespace std;

class Complex {

private:

int real, img;

public:

// Constructor

Complex(int r = 0, int i = 0) : real(r), img(i) {}

// Getter and Setter

void setData(int r, int i) {

real = r;

img = i;

}

void getData() const {

cout << real << " + " << img << "i" << endl;

}

// Overload Operators

Complex operator=(Complex c) {

real = c.real;

img = c.img;

return \*this;

}

bool operator==(Complex c) {

return (real == c.real && img == c.img);

}

Complex operator+(Complex c) {

return Complex(real + c.real, img + c.img);

}

Complex operator++() {

++real;

++img;

return \*this;

}

Complex operator--() {

--real;

--img;

return \*this;

}

// Function to display complex number

void showData() const {

cout << real << " + " << img << "i" << endl;

}

};

int main() {

Complex c1(3, 2), c2(1, 1), c3;

c1.getData();

c2.getData();

c3 = c1 + c2;

c3.showData();

c1;

c1.showData();

c2;

c2.showData();

if (c1 == c2)

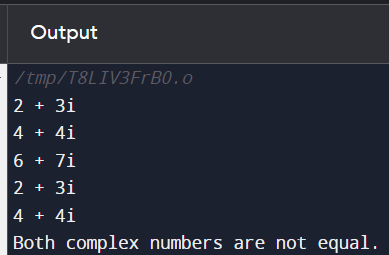
cout << "Both complex numbers are equal." << endl;

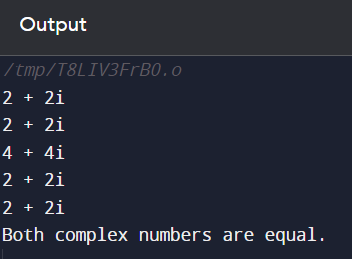
else

cout << "Both complex numbers are not equal." << endl;

return 0;

}





4. Write a C++ program to overload ‘!’ operator using friend function.

5. Read a value of distance from one object and add with a value in

another object using friend function.

Code:

#include<iostream>

using namespace std;

class Distance {

private:

int meter;

public:

Distance() : meter(0) {}

friend int addDistance(Distance, Distance);

};

int addDistance(Distance d1, Distance d2) {

return d1.meter + d2.meter;

}

int main() {

Distance d1, d2;

int dist1, dist2;

cout << "Enter the distance of the first object: ";

cin >> dist1;

d1.meter = dist1;

cout << "Enter the distance of the second object: ";

cin >> dist2;

d2.meter = dist2;

cout << "The sum of the distances is: " << addDistance(d1, d2) << endl;

return 0;

}

